Massachusetts Institute of Technology
Lincoln Laboratory

A Study of Gaps in Attack Analysis

Hamed Okhravi
Thomas Hobson
Chad Meiners
William Streilein

Technical Report 1193

October 18, 2015

This work is sponsored by the NSA's Cyber Defense Research & Technology Team under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government. Distribution A: Public Release

Lexington Massachusetts

TABLE OF CONTENTS

		Page
	List of Tables	v
1.	EXECUTIVE SUMMARY	1
2.	INTRODUCTION	3
2.1	Goals	3
2.2	Scope	3
3.	METHODOLOGY	5
3.1	Threat Model	5
3.2	Gap Discovery Process	6
3.3	Gap Selection Process	6
3.4	Gap Treatment and Classification Process	6
3.5	Gap Prioritization Process	6
4.	GAPS AND RESEARCH DIRECTIONS	9
4.1	Overview of Root Causes	9
4.2	The semantic gap between the attacks and detection mechanisms	10
4.3	Systems are designed for expressiveness and flexibility	14
4.4	Production and analysis environment asymmetry	24
4.5	Legacy support requirements	29
4.6	Sensors are static spatially and temporally, while attacks are dynamic	29
4.7	Loose Tying between Identities and Entries over the Network	32
4.8	Enormous volume of unknown activities (mostly benign)	35
4.9	Underdeveloped reversible and low impact defenses	37
4.10	Systems implicitly or explicitly trust the users	38
5.	SUMMARY OF RESEARCH DIRECTIONS	41
	References	45

This page intentionally left blank.

LIST OF TABLES

Table No.		Page
1	Threat Matrix of study	5
2	Qualities of criteria	7
3	Overall Gap Prioritization Table	43

This page intentionally left blank.

1. EXECUTIVE SUMMARY

The ability of the defender to detect and identify cyber attacks reflects the "arms race" nature of the cyber domain. While defenders develop new and improved techniques to detect known attacks, attackers resort to more sophisticated and stealthy techniques to perform their intrusions and evade detection. In this study, we identify the major gaps that exist in today's attack detection systems and infrastructures that impede more efficient and effective attack analysis. Attack analysis in this study refers to activities related to identification and understanding of attack methods and techniques, the capability to detect such attacks in USG, DoD, and general enterprise systems, the ability to attribute such attacks to adversaries, and the ability to predict them before they happen. Since the latter two capabilities are significantly underdeveloped, most of the focus of this study is on the identification and detection of attacks. We have reviewed recent literature and available to identify major gaps in attack analysis. We have then ranked these gaps based on their likelihood of impacting current systems, the extent of their impact, and the cost of developing new and improved techniques to improve current attack analysis capabilities. We have ranked higher those gaps that have higher likelihood, higher impact, and lower cost. We have then identified promising research directions that should be pursued to mitigate these gaps. For most gaps, we have identified a short term (1-3 years), a medium term (3-5 years), and a long term (5+ years) research direction. Our research recommendations fall into seven high level directions: incorporating population-based attack analysis as opposed to analyzing individual attacks, making detection sensors temporally and spatially dynamic, making attack identification and detection more collaborative among systems and enterprises (e.g. across DoD and USG), applying more automation to currently manual techniques (e.g. automated signature development), using multi- resolution analysis that puts more focus on the more prominent attack techniques and sensitive parts of a system, developing more relevant, accurate, and representative corpora of attacks, and incorporating existing hardware extensions to support better and more accurate detection of attacks. We do not have enough evidence to believe that the list of gaps and research directions identified is complete or comprehensive. However, we have made our best attempt to identify the major gaps in attack analysis using our knowledge and judgment and we strive to help and guide the research directions with this report.

2. INTRODUCTION

This report provides a gap analysis for the area of attack analysis. For the scope of this paper attack analysis is defined as techniques and capabilities to discover, identify, and prevent adversary activities on enterprise networks; these technique address weaknesses in the ability to detect, notify, and react to cyber attacks.

2.1 GOALS

This report is designed to gather together gaps in the area of attack analysis from open sources such as top tier networking and security conferences. It is the intention of the authors that this report be shared with all members of the community in order to foster research lines that will address the gaps identified within this report.

2.2 SCOPE

The information within this report is entirely derived from open sources. In order to make the report applicable to the largest audience we have not considered proprietary environments or information sources. We make the general assumption that the domain requiring protection is that of a typical large enterprise. We assume such enterprises have adequate resources to fund an internal cyber defense team.

The remainder of the report is organized as follows. The second section describes our study methodology, which includes the threat model that we use for this study as well as our method for finding, selecting and evaluating gaps for inclusion in the report. The third section describes all gaps that we have selected for inclusion, organized by the primary root cause of the gap. In the third section we also describe each gap; we describe the gap's impact; and we discuss lines of research that may address the gap.

This page intentionally left blank.

3. METHODOLOGY

In this section we describe our methodology for gap discovery, selection, treatment and prioritization.

3.1 THREAT MODEL

The threat model for enterprises has two dimensions: who is attacking and why they are attacking. Table 1 illustrates the capability/desire levels for actors engaging in activities against an enterprise. This capability/desire is qualified as low, medium or high because there are other factors that contribute to attacker activities such as a threat actors interest in a particular enterprise, which would help determine the actual threat to a specific enterprise. As such, this matrix is used to scope the broad capabilities and interests and additional information would be used to determine actual threat for a specific enterprise.

TABLE 1

Threat Matrix of study

Tier 1	Billions of assets	🟥	🟩	🟩
Tier 2	Millions of assets	🟨	🟩	🟩
Tier 3	Thousands of assets	🟩	🟨	🟨
Threat Actor	Description	short term profits	Long term profits	Mission based effects
Low Medium High	Threat Activity	Criminal	Espionage	Warfare

Table 1 illustrates the trend that actors with larger amounts of assets focus primarily on cyber-espionage and cyber-warfare. This makes logical sense because well-funded threat actors are primarily nation state actors, or large organizations, which means that they will use cyber activities as a means to accomplish their internal mission objectives. Low funding threat actors primarily focus on criminal based activities although they may also engage in cyber-espionage and cyber-warfare if they are suitably motivated into action. For the scope of this report we primarily address gaps for which threat actors with low to medium levels of funding would have the capability to exploit. This decision provides for the greatest lower common denominator for gaps that need to be addressed by enterprises.

3.2 GAP DISCOVERY PROCESS

To discover gaps in the attack analysis capabilities, we conducted a literature survey of over sixty papers from top tier networking and security conferences such as IEEE Security and Privacy, USENIX Security, ACM Conference on Computer and Communications Security (CCS), and the Network and Distributed System (NDSS) Symposium, Symposium on Research in Attacks, Intrusions, and Defenses (RAID), Annual Computer Security Applications Conference (ACSAC), among others.

We initially surveyed the titles and abstracts for the past four years of each conference's proceedings. We then down-selected this very large list of papers based on their relevance to attack analysis. We read the papers from this refined list and extracted any relevant gaps based upon our own experience and the context of the paper and the state research contribution.

3.3 GAP SELECTION PROCESS

The gap selection process was performed during team meetings and began with individual team members independently determining if a discovered gap was worthy of carrying forward. A majority vote was then used to determine whether to keep it as a gap or not. Also, during this stage, duplicate and similar gaps where merged into single gaps.

3.4 GAP TREATMENT AND CLASSIFICATION PROCESS

Gap treatment and classification was initially done independently by each team member. The process involved going through each gap and devising a root cause for that gap's existence. The root causes were clustered into at most four root causes per team member. These root causes were discussed and combined into nine root causes via discussion, review, and majority agreement. After the root causes were agreed upon, team members classified each gap by determining its primary root. By primary root cause, we mean the root cause that contributes the most to the existence of the gap. We also assigned secondary root causes to gaps that have more than one cause. The list of root causes can be found in the next section, Gaps and Research Directions; gaps have been organized according to primary root cause.

3.5 GAP PRIORITIZATION PROCESS

Gaps have been prioritized according to three criteria: *likelihood*, *impact* and *cost*. *Likelihood* is the likelihood that the gap is being used by threat actors in today's systems to bypass defenses or to avoid detection. *Impact* is the damage the gap leads to in enterprise systems when exploited by an attacker. *Cost* is the cost of fixing the gap. All criteria were given qualitative values of low, medium and high as there are no reliably objective metrics that are simple enough to calculate accurately within the constraints of this study. Table 2 lists qualities that we used to decide the values of each gap's criteria.

Each team member individually scored each gap's criteria. The scores for each gap where consolidated by majority agreement to produce a master score for each gap's criteria. Once a master score was determined, a priority with the equation $C \times \sqrt{L \times I}$ where

$$C = \begin{cases} 1 & \text{if gap cost is high} \\ 2 & \text{if gap cost is medium} \\ 3 & \text{if gap cost is low} \end{cases} \qquad L, I = \begin{cases} 1 & \text{if likelihood, impact cost is low} \\ 2 & \text{if likelihood, impact cost is medium} \\ 3 & \text{if likelihood, impact cost is high} \end{cases}$$

With this priority ranking, we determined the priority of each gap. Note that our priority ranking is a subjective computation based on the qualitative ranking of our team members. Other researchers are likely to compute different priorities based on their scoring of the gaps.

TABLE 2

Qualities of criteria

	Likelihood	Impact	Cost
High	We see this problem every day	It can severely limit our defenses or damage our systems	We have to redesign and change many systems and infrastructure
Medium	We see this problem in some systems	It has negative impact on our systems	It requires research and deployment efforts
Low	We barely see this problem	It can have negative impact, but the impact is small compared to other gaps	The systems and infrastructure remains mainly unchanged; the development is possible

This page intentionally left blank.

4. GAPS AND RESEARCH DIRECTIONS

4.1 OVERVIEW OF ROOT CAUSES

Nine root causes have been identified as the major sources of the gaps. Each of these root causes is hard to fix in its entirety, however, even partially addressing a root cause has the potential to impact many gaps. While the ultimate goal is to discover solutions for these root causes, researching how to improve individual gaps may provide more immediately tangible results. Gaps are described under the root causes from which they primarily stem. Given that many gaps are attributable to multiple root causes, secondary root causes are discussed within the gaps as applicable. Here we provide a brief description of the nine root causes and expand upon these causes in the discussion of each gap that follows below.

1. **The semantic gap between the attacks and detection mechanisms**
    a. There is often a limited understanding of the actual meaning of data at a point of detection or analysis, typically due to a lack of context from the points of attack.
2. **Systems are designed for expressiveness and flexibility**
    a. Many languages, protocols, and system interfaces are designed to be expressive and flexible in order to facilitate usability expandability and general applicability. Attackers can similarly take advantage of these properties, complicating attack analysis. Furthermore, benign but vulnerable systems become harder to analyze when they are less constrained.
3. **Production and analysis environment asymmetry**
    a. Operational environments have proven incredibly challenging to replicate. This complicates detecting and analyzing attacks that appear in specific environments as well as testing and training defenses in realistic environments.
4. **Legacy support requirements**
    a. Solutions for some gaps are known today yet existing infrastructure often cannot be easily upgraded for economic, political, group coordination, or other reasons and thus gaps remain unresolved.
5. **Sensors are spatially and temporally static, while attacks are dynamic**
    a. Attacks are comparatively more dynamic than sensors, changing targets, sources, and payloads. This is done in order to evade detectors that rarely change in terms of content searched as well as positioning within the network, system, and software stacks.
6. **Loose tying between identities and entities over the network**
    a. The inability to associate an individual or group with an entity has long been pervasive throughout network communications and substantially limits attribution. Additionally, the inability to track a suspicious actor over time hinders detections based upon trends.

7. **Enormous volume of unknown activities (mostly benign)**
   a. Network communications and system activity continually generate large volumes of data which limits the potential analyses that can be performed and exacerbates the problem of managing false positives.

8. **Underdeveloped reversible and low impact defenses**
   a. Detection technologies have few options for responsive actions with low impact on the network or that can be undone with relative ease (e.g. putting spam in junk folder). Lack of such options limits the utility of detectors with false positives.

9. **Systems implicitly or explicitly trust the users**
   a. Many systems have been built to trust users due to both intended design and errors in implementations. While this has improved usability overall it has also caused challenges for attack analysis, specifically in regards to the ease with which attackers can mask activities, identities, and capabilities.

4.2 THE SEMANTIC GAP BETWEEN THE ATTACKS AND DETECTION MECHANISMS

Gap: Missing end-to-end view required for understanding attacks involving multiple parties

Likelihood: H Impact: H Cost: L

Description

The community has provided few options for analyzing attacks that span across multiple parties, leaving a gap in piecing together the full view of attacks. This gap is particularly salient in the web server and browser domain where there may be just two parties, the site and the user. Consider Cross-Site Request Forgery (CSRF) attacks in which a browser visiting a malicious site could issue a fake request to a victim site on behalf of a user [9, 36]. The victim site does not know that the request came from the malicious site as opposed to the user, it simply sees the request as coming from the same browser. The browser does indeed possess information that could enable the server to make this decision and could share this information with the server, perhaps via an HTTP Referer header, but many browsers avoid this for privacy or other reasons. Lacking an end-to-end view prevents the detection of attacks for both isolated incidents and higher-order, coordinated attacks. It also limits forensic investigators analyzing attacks after the fact.

The primary cause of this gap is that the system making the detection or security decision lacks complete information, information which may actually be known by other parties. A major reason that we do not have the ability to provide this information is because we lack the appropriate sensors and mechanisms for extracting data from those sensors at decision points.

Research Direction

In the short term focus should be directed towards developing sensors that are sharable across client-server interactions, where there are solely two parties, each of which possesses part of the picture. Attention should be given to connecting simple, existing sensors on the client and server sides to better detect web-based attacks. For example, browser-based taint tracking [88] can be combined with server-side traffic monitoring [43] to better detect XSS attacks. Additionally, we need ways to facilitate collaboration amongst cooperative parties. Several collaborative detection systems have been proposed in the academic community that can serve as starting points [13, 97].

In the medium term seek to integrate sensors into systems that are owned and operated by external parties. This might involve placing sensors in uncooperative areas and procuring services solely for the purpose of feeding attack-relevant information. Work described in [3] has demonstrated such an approach in their investigation into domain parking services. In order to identify a domain parking service as malicious it was necessary to host web sites with the parking service, run advertisements via third-party ad networks, purchase traffic from third-party providers, and simulate web crawling activities. Detecting malicious behavior required sensors at all of these layers.

Longer term efforts should develop capabilities that enable dynamic sensing across parties. While a detection system may not know how to properly make a security decision it will often recognize when it does not have sufficient information. Lacking the ability to dynamically poll other sensors for the missing data, current sensors simply stop the analysis here. Research can focus on selectively and randomly enabling a large set of such sensors [14] (which cannot all be active at all times due to performance overhead) in order to achieve a more diverse and randomized view of networks. Advancing this capability may also involve research into privacy-preservation techniques, supporting the establishment of partnerships amongst relevant parties, and development of an application introspection interface (ASI) that enables each application to provide a standardized semantic view of an application similar to how APIs provide a standardized programming interface.

Gap: Network stack manipulation cannot be found by signature matching

Likelihood: H Impact: M Cost: M

Description

Network protocols are very complex and flexible. Many detection techniques use signature-based schemes to detect network stack manipulation attacks similar to malware detection. However, this has been a losing battle. For example, attacks that manipulate a higher layer protocol field (e.g. FTP) cannot be easily detected at the network layer by inspecting packets. First, analyzing protocol manipulations often requires emulating the protocol which is difficult and performance heavy. Stateless approaches such as signature matching miss the rich semantics of protocol manipulation attacks [51]. Second, protocols often provide many different ways of achieving a desired behavior which enables an attacker to evade signature detection by making changes to the implementation of an attack while maintaining the malicious effect.

The main root cause of this gap is the semantic gap between the detection logic (signature matcher) and the attack (protocol manipulations). Another contributor to this problem is the requirement to support legacy network protocols. Many of these complexities can be avoided if the entire protocol stack can be designed from a clean slate.

Research Direction

Short term research should focus on designing a few detection techniques built into the same network layer as the most prominent types of protocol manipulation attacks. For example, research should focus on high level protocol analyzers to detect botnets [33].

Medium term research should focus on limiting the flexibility of existing protocols by disabling their unused features or forcing them to behave in a canonical way to make them more amenable to detection [58].

Long term research should focus on development of new network stack protocols that can be provably secure and their misuse can be detected using simple and lightweight approaches.

Gap: Network flow level metrics are insufficient to accurately detect botnets

Likelihood: H   Impact: M   Cost: M

Description

Network operations centers commonly use network flow data in order to detect malicious bots on their network. Some information about network connections can be understood at the network flow level. This typically includes the communicating host addresses, transport-layer communication protocols, sizes and number of packets, and connection times. Unfortunately, analyses at the network flow level commonly miss even relatively loud botnet communications. The limited number of attributes that we can check at the network flow level [78] allows attackers to easily appear benign when measured under these attributes. Attackers simply ensure malicious behavior is only visible at a finer granularity. For example, packet payloads are not visible at the network flow level and thus a bot could be receiving unencrypted command data from a botnet master and a network flow level analysis would have no visibility into these commands.

This is a gap area for nearly all organizations and most directly impacts network analysts who are attempting to detect both existing and new malicious behavior at a network-level view. This also inhibits research into network defenses in general; researchers have had limited success using the few attributes that can be analyzed at the network flow level.

The root cause is that there's a semantic gap: we don't have the context at the network flow level that's required for understanding the malicious communication. We don't have information about packet payloads or encryption keys that might only be available at the host or application level but which would provide more insight into the nature of the communication. Another cause for this gap is the fact that many network detection approaches largely ignore installing or using sensors that could provide more details and instead rely heavily upon sensors (i.e. network flow producers) that can capture more manageable volumes of data.

Research Direction

In the short-term, research can look at more effective ways to use network flow level information for detecting botnets, such as the work described in [10] that uses a machine learning approach to detect botnet command and control servers. However, given the limited amount of information available at this level, this will likely only provide incremental improvement. In the mid to long term the community should consider moving more towards building networks and systems that are capable of deeper sensing. This could come in the form of more dynamic sensors that only capture detailed, large volumes of data when triggered, either by an event or periodically. For example, observing network flows with suspicious connections to one server might trigger sensors on the clients to begin logging full packet capture data.

Gap: Lack of effective techniques for tracking input sources through memory and storage

Likelihood: M Impact: H Cost: M

Description

A gap exists in the ability to track malicious input as it moves within a program's memory space (and potentially to storage or other systems). The typical use case is to detect input that's intended to hijack control flow or change sensitive data. This could be input designed to overwrite a return address, instructions or a command to be executed, or malicious changes to a data structure. Tracking input in the context of a software application is referred to as taint tracking [62, 76]. Closing this gap would enable detectors to distinguish attacker supplied values from other program data and thus detect if an attacker supplied value were to change sensitive data such as a return address.

The gap is of interest to researchers building detections, mainly with the goal of protecting against memory corruption vulnerabilities but also of protecting against higher-level attacks including SQL injection, command injection, or general malicious data modification. It is also of interest to reverse engineers who are attempting to understand an attack and identify the vulnerability that was exploited by the malicious input. Those wishing to track adversarial movements and persistence on a network are also hindered by this gap as they have trouble distinguishing adversarial data from benign data.

The semantic gap is the root of this problem. At the point in which a security decision must be made about a piece of data most or all of the information about the source of this data has been lost. This is closely tied to the fact that languages and systems are expressive and flexible. With an enormous number of potential program paths, the challenge lies in retaining source information throughout execution in a performance efficient manner.

Research Direction

Investigate ways to improve the performance of taint tracking for small software applications. The overhead of dynamic taint trackers is currently very high, typically over 100% [81]. One of the faster taint tracking prototypes to date is Minemu [15]. For more practical applications of input tracking look at the prototype described in [23] which presents a technology that uses taint and data flow analysis to detect vulnerabilities in PHP web apps. In the medium term hardware solutions should be investigated to determine the feasibility of bringing performance to reasonable levels. Longer term seek to move beyond taint tracking and following input within a single piece of software and towards researching how to track input through to storage and other systems. Such research would support tracking adversarial movements on a network.

Gap: Most protocols are binary and have high variability; N-gram anomaly detection is not an effective technique against binary protocols

Likelihood: L Impact: L Cost: H

Description

N-gram anomaly detection has been proposed to detect attacks in different contexts. However, N-gram anomaly detection has had limited effectiveness against network protocols. Since most protocols are binary, N-gram analysis misses a lot of semantic information (e.g. protocol fields such as addresses, headers, flags, etc.) that's necessary for creating detection technologies that are hard to evade and have low false positive rates [34].

The main root cause of this gap is the semantic gap between the detection algorithm and the attack. Protocols use fields, addresses, and flags whereas N-gram analysis looks at the binary representation of such values without that semantic information. Another contributor to this gap is the legacy support for existing network protocols.

Research Direction

Medium term research should focus on anomaly detection approaches for a limited number of popular protocols, but by focusing on protocol fields as opposed to inspecting binary byte values. Another possible direction is disabling unused features of existing protocols to make them more analyzable [58].

Long term research should focus designing new network protocols that are more amenable to analysis via anomaly detection.

4.3 SYSTEMS ARE DESIGNED FOR EXPRESSIVENESS AND FLEXIBILITY

Gap: Static analysis techniques are hindered by packing and obfuscation techniques

Likelihood: H Impact: H Cost: L

Description

Malware payloads are often packed or obfuscated in order to evade detection and reverse engineering. Recent studies have estimated that between 50-92% of malware is packed [1, 17]. This complicates the network defense and triaging process because the analysts have to spend a considerable amount of time understanding and unpacking the malware's behavior. This also impedes automated ways of analyzing the malware's impact [17, 45, 67]. One implication of this gap is the difficulty it creates in prioritizing malware investigations and defenses. This problem also hinders classification, taxonomy generation, malware labeling, and information sharing. In fact, it has been shown that the majority of anti-malware/anti-virus vendors disagree on malware labeling for most malware samples [67]. The flexibility and expressiveness of the current processor architectures (such as x86) and operating systems is the root cause of this problem. The fact that there are many ways of performing an operation in modern systems allows malicious behavior to be obfuscated which makes static analysis very hard. Backward compatibility (i.e. legacy support) forces outdated features to remain available, providing yet more ways for a payload to be obfuscated.

Research Direction

Short term research that can mitigate this gap is to dynamically extract malware behavior from a population of known malware and statically check those specific behaviors against an unknown sample [45, 46, 50].

Medium term research should focus on sharing analysis reports across systems and networks so that if the malware exposes the hidden behavior on a system, it can be used in other systems and networks to understand its impact [40, 50, 57].

Long term research can focus on designing new architectures or tagging existing ones with additional mechanisms that allow clean separation of code and data. This will remove flexibility for the attacker in his ability to achieve certain behaviors [14, 88, 92].

Gap: Lack of effective techniques for detecting code injection

Likelihood: H Impact: H Cost: M

Description

Despite many years of research into determining if a given sequence of instructions is malicious, it has proven challenging to implement effective detection mechanisms and other defenses in practice. Memory corruption vulnerabilities continue to be one of the most common ways that malicious code is executed on victim systems [40, 81]. This gap specifically relates to challenges associated with malicious code injection while a subsequent gap (see Gap: Lack of effective techniques for detecting code reuse) relates to the challenges associated with protecting against attacks such as Return-Oriented Programming [47, 68]. By code injection we refer to the insertion of malicious instructions into an application by an attacker and the hijacking of control flow in order to execute those instructions. This gap impacts software written in languages that lack memory safety, notably C and C++. Techniques such as W xor X [59, 82], and to a lesser extent code signing in Apple's systems [65], have proven to be effective at preventing injected code from being executed. However, it is not always possible to implement these techniques in all software, specifically in software using Just-In-Time compilation [7] and in cases where it can be disabled by code reuse attacks.

The ability to definitively detect the injection or execution of malicious instructions would eliminate some memory corruption attacks entirely and make most attacks harder to accomplish. In the case of perfect detection of code injection, attackers can sidestep code injection detection by reusing existing code in the application, however the difficulty of attacks would undoubtedly be raised; attackers would have to complete attacks entirely by reusing code, which requires more effort than simply injecting code. This gap stems from the fact that we develop software in languages that lack memory safety. Applications are developed in such languages in order to take advantage of the flexibility and speed provided by these languages. In cases where developers would consider switching to memory safe programming languages we would still have to consider supporting a large, complex, and widely used code base that is written in C and C++ (e.g. most operating system kernels and system-level software).

Research Direction

Performance and compatibility have been the two major reasons that more memory corruption defenses have not been implemented to date. Upcoming hardware implementations of proposed defenses have the potential to significantly improve the performance of these defenses, which have traditionally been hindered by slow software implementations. Efforts in the near term should be directed at supporting upcoming hardware implementations. Specifically, consider how the new Intel Memory Protection Extensions [31, 39] can be used to achieve memory safety in commodity applications and explore potential hurdles to successful adoption.

In the medium term it would be useful to investigate other hardware or performance-efficient software solutions to address any shortcomings identified in the Intel Memory Protection Extensions. An alternative approach may be a hardware implementation of Instruction Set Randomization [8, 47, 68], which provides a unique encoding of instructions such that unencoded instructions injected by an attacker would fail to execute. Another medium term research direction should be the practical applications of Trusted Computing extensions [2, 25, 31] which verify the authenticity of any code running on the machine before it is executed. Practical challenges such as managing large lists of valid software hash values [65] and the difficulty of applying patches in the presence of Trusted Computing defenses have prevented their large-scale deployment to date. By addressing these challenges, Trusted Computing techniques can provide a strong defense against code injection. This is especially true given the fact that the necessary hardware modules such as the Trusted Platform Module [65] already exists on a large number of today's systems.

Longer term, consideration should be given to research that helps migrate programmers to type safe languages. Typed Assembly Languages exist [60] and operating systems have been developed in type-safe languages [95] but they have not generated much interest from the systems development community. Research is needed to determine how we can practically perform low-level system programming in safer languages, improve usability for programmers, and aid in migrating existing code bases.

Gap: Lack of effective techniques for detecting code reuse

Likelihood: H Impact: H Cost: M

Description

In addition to code injection attacks (see Gap: Lack of effective techniques for detecting code injection), memory corruption attacks enable a more dangerous and harder to detect form of code reuse attacks [18, 19]. In code reuse attacks, attackers do not need to inject malicious code into the victim's memory; rather, they reuse code that already exists in memory from the application's code and linked libraries in order to achieve malicious behavior. In its simplest form, the attacker can directly jump to a sensitive function [91]. In its more advanced form, the attacker can stitch together short pieces of code (often called gadgets) to achieve arbitrary behavior on victim's machine [19]. With the advent of many defenses against code injection, code reuse attacks have become a widely popular and easily accessible tool in an attacker's arsenal [20, 93]. Many current attacks incorporate a code reuse stage in order to bypass code injection defenses [20]. These attacks are much more challenging to detect because traditional defenses against code injection attacks such as Trusted Computing techniques [2, 31], code signing [65], or any technique that focuses on detecting foreign code on a machine does not work in stopping them. Many recently proposed defenses against code reuse attacks such as detecting their distinct features or defenses based on detecting anomalous control flow have been shown to be ineffective [37, 54, 64, 75]. As code reuse techniques and tools are readily accessible to attackers [38] and modern operating systems have somewhat better defenses against code injection attacks [84], it is expected that code reuse attacks will continue to grow in significance in the coming years.

The root causes of this gap are very similar to those for code injection attacks. Lack of memory safety in most popular programming languages and vast codebases that are developed in such languages contribute to this problem.

Research Direction

Short term research should focus on two thrusts. For new software applications, compiler-inserted checks [25] must be combined with lightweight forms of memory randomization [37] to detect and prevent code reuse attacks. The combination of these existing approaches can significantly raise the bar for such attacks. For existing software applications, simple hardware extensions such as a tagged architecture [2, 92] and Intel's SGX [6] should be evaluated and deployed to detect code reuse attacks.

Medium term research should focus on hardware assisted randomization and decoys to detect code reuse attacks.

Long term research should focus on using memory safe languages or augmenting memory unsafe languages with efficient and lightweight features and checks that make them memory safe [39, 42]. Existing techniques fail in this front because they incur very high overhead [25].

Gap: Lack of basic understanding of information leakage attacks

Likelihood: M Impact: H Cost: L

Description

Memory corruption bugs are one of the oldest problems in computer security [81]. One particularly dangerous type of memory corruption attacks is information leakage (i.e. memory disclosure attacks) that has been used widely to bypass existing systems security defenses such as memory randomization, or stack protections [47,68]. This type of attack has become even more common in recent years in exploits and malwares [8]. There is a severe lack of tools, techniques, and understanding of the mechanisms and impact of information leakage attacks in the community and they continue to pose significant threats to existing attack detection capabilities.

The root causes of this gap are very similar to those for code injection attacks. Lack of memory safety in most popular programming languages and vast codebases that are developed in such languages contribute to this problem.

Research Direction

In short term new techniques such as code property graphs [94] can be used to analyze and identify this type of vulnerability. In medium and long term, more in depth analyses of the impact of information leakage on adversarial actions is necessary [31].

Gap: Approaches based on syntactic information do not work; approaches based on semantic information are understudied

Likelihood: H Impact: H Cost: M

Description

The vast majority of attack detection approaches to date have relied on specific attack features (i.e. syntactic information). A special case of this gap is the antivirus community's focus on signature-based malware detection, but the gap is a wider problem and has been seen in other communities as well. For example, detection of code reuse attacks based on an anomalous number of return instructions [47] (its weaknesses [67]), detection of control hijacking attacks based on tracking sensitive system calls [68] (its weaknesses [8]), detection of malicious code based on its byte representation [65] (its weaknesses [31]), and detection of malicious nodes based on IP addresses (its weaknesses [2]) are all examples of using syntactic information for detection (i.e. return instructions, certain system calls, non-printable characters, and IP addresses). The problem with syntactic information is that it is not intrinsic to how the attack works (i.e. semantic information). Since there are many ways of achieving a behavior in modern systems, syntactic information can be modified by an attacker to bypass detection capabilities based on such information. Polymorphic malware and the weaknesses associated with syntactic-based detection demonstrate this gap.

A natural step would be to focus on semantic information, that is, features of attacks that are intrinsically tied to what the attack does and cannot be modified without breaking the intent of the attack. However, such techniques have been understudied and are less practical. The difficulties in semantic-based techniques arise from multiple factors. First, the intrinsic behavior of an attack is not always understood. Second, notions of malicious vs. benign often rely on higher level contexts.

Third, even if the problem is well-defined, enforcing semantic-based detection often incurs a high false positive rate.

The main root cause of this gap is the flexibility and expressiveness of the modern systems in which many different valid syntactic representations exist for a given semantic behavior. Moreover, the large volume of benign activities makes many semantic-based techniques impractical because of false positives.

Research Direction

Short term research should focus on automatically extracting semantic information from a large population of attacks, specifically for malware analysis. Techniques such as the one proposed by Fredrikson et al. [27] are the first steps in that front. Moreover, research purely based on syntactic information or features that are not intrinsic to how an attack works should be curbed because such techniques have been shown to be weak.

Medium term research should focus on characterizing semantic features of large classes of attacks (e.g. malwares, web-based attacks, code reuse attacks, etc.) and on easily reversible quick reaction techniques to mitigate the impact of false positives. Note that if the actions taken based on a false detection are easily reversible, it would potentially allow tolerating higher false positive rates in practice.

Long term research should focus on developing processor architectures and operating systems that can be configured to be less expressive and more special-purpose [6]. By tailoring systems toward a special purpose, the effectiveness of syntactic-based detection can be improved since there will be fewer ways of performing an operation. This can also improve the accuracy of semantic-based detection approaches by reducing the number of false positives.

Gap: Measures of attack surface and vulnerability scores are inaccurate and arbitrary

Likelihood: H Impact: M Cost: M

Description

There have been many attempts to quantify and define metrics for attack surfaces [56] or vulnerabilities [74]. Unfortunately has been little attempt to verify the validity or correctness of such measures. To the contrary, empirical studies of attack surfaces and vulnerability exploitation in the wild actually indicate that existing metrics are inaccurate and arbitrary [61].

The main root cause of this gap is that existing metrics quantify specific ways of attacking systems, whereas in reality there are numerous ways of attacking systems because of their flexibilities. Moreover, because the production environments in the wild do not behave like analysis ones, subjective metrics can often prove inaccurate in practice.

Research Direction

Short term research should focus on defining new metrics for vulnerability scores and attack surfaces based on observed exploitation of vulnerabilities in the wild [61].

Medium term research should focus on developing targeted detection techniques that do not treat all attacks and vulnerabilities the same; rather, they incorporate ground truth data collected in the wild to detect different attacks with varying sensitivities and adjustable parameters.

Long term research should focus on automated scoring and prioritization systems based on observed traffic and detection events. Large-scale information sharing would be a necessary component of such an automated system.

Gap: Anti-virus systems provide an oracle to malware writers seeking to evade detection

Likelihood: H   Impact: M   Cost: H

Description

Anti-virus (AV) systems rely heavily upon signature-based detection to detect malware. This has numerous problems such as ease of evasion. One of the fundamental problems of AV systems is that they can be used as an oracle for malware writers to evade detection [83]. They can automatically re-pack and obfuscate a malware sample until it cannot be detected by any AV system. In fact, popular services such as VirusTotal [83] can be used for such purposes.

The main root cause of this gap is that systems are flexible and expressive enough to allow easy perturbations to malware that achieve the same behavior, but can avoid detection.

Research Direction

Short term research should focus on randomized and hard to reverse engineer detection systems that cannot deterministically be evaded, but this would only mitigate this gap to some extent.

Medium term research should focus on sharing analysis reports across systems and networks so that if the malware exposes its unobfuscated behavior on a system, it can be used protect other systems [36].

Long term research can focus on designing new architectures or tagging existing ones with additional mechanisms that allow clean separation of code and data and is less flexible for achieving certain behaviors [6, 92].

Gap: Dearth of techniques focused on targeted attacks

Likelihood: M   Impact: H   Cost: H

Description

The characteristics of targeted attacks are often dissimilar to those of widespread attacks. For instance, phishing attacks may be more directed, adversaries may make more frequent use of zero-days, or stealth may play a larger role [12]. Despite the fact that targeted attacks have unique characteristics, current commodity detection systems and the majority of research are directed at more general attacks. This potentially leaves a gap for network analysts at all organizations but most notably those involved in government, critical infrastructure and services, technology, activist groups, and sector leaders.

A cause for this gap is that all networks are unique by design, enabling adversaries to craft custom attacks that incorporate unique properties of the target, properties that are often outside the scope of detection systems. Spear phishing provides one example where an adversary can craft an email using unique characteristics about the target, specifically custom content and names. Simply avoiding the more general and widespread attributes of traditional spam emails renders existing detectors ineffective. Another cause is that the systems on which we test our technologies often exclude many unique elements of the target network. This is particularly challenging in the research community where experiments are encouraged to be as generic as possible.

Research Direction

The community must first establish the requirements for detection and analysis technologies that are capable of incorporating targeted information. The research community has begun to provide analyses and threat models for targeted attacks, which can serve as a foundation for better understanding the unique characteristics and requirements for new target-focused technologies [12, 21, 35, 69, 83].

Entities must also begin to develop custom defenses, something that is rarely done in practice today. Given the costly nature of these initial custom defenses, this should serve as more of a medium term solution with a main objective of refining requirements for technologies that incorporate network-unique aspects. Longer term, researchers should work to find more cost-effective ways to tailor defenses to each entity.

Gap: Techniques to increase coverage suffer from path explosion

Likelihood: M Impact: M Cost: M

Description

Software analysis approaches, notably symbolic execution, are limited by the path explosion problem [5, 48]. The path explosion problem refers to the fact that the number of paths of execution grow exponentially with the size of a program, making it infeasible to execute all paths for large programs. Those attempting to build automated software analysis tools as well as malware analysts are limited by this gap. It prevents analysts from detecting vulnerabilities in software and from finding dormant behavior in malware [73].

The root cause for this lack of coverage is that our systems are designed to be expressive and flexible, thus operating on a wide range of input in a wide range of states, all of which need to be covered for a complete analysis. In terms of finding dormant behavior in malware, the flexibility and diversity of execution environments creates the potential for a vast number of program states in which the malware can choose to hide its malicious behavior.

Research Direction

In the short term research is needed to develop approaches and heuristics that look to achieve high coverage in limited but critical areas of programs. Recognizing that the path explosion problem persists after decades of research in this field [5], we should accept that the problem is likely to persist in the near term and focus our efforts on improving coverage for higher risk code rather than the entirety of large programs.

In the medium term, efforts should investigate ways to improve the parallelizability and overall performance of path exploration. Longer term research is needed to create more efficient constraint solvers, which are used for determining which input values are required for reaching given paths.

Gap: Unable to generalize detection of malicious behavior

Likelihood: H Impact: H Cost: H

Description

Detecting malicious behavior is largely limited to searching for specific known malicious properties, often as simple as exact byte pattern signatures. Unfortunately narrow indicators can easily be evaded by malware via the use of syntactically different but semantically equivalent code [27,55,73]. A substantial challenge in generalizing our detections lies in keeping false positives sufficiently low. Generalizations that are too broad often encompass benign code as well. Closing this gap would enable the detection not just of instances of known malware as is common today but also of new malware. It would also reduce the ongoing and substantial maintenance effort required for creating signatures. The gap is particularly relevant to analysts responding to many false positives while remaining blind to false negatives from new attacks. Malware analysts would also benefit from closing this gap. Malware analysis efforts could be drastically reduced if analysts were able to recognize malicious behavior in one sample and correlate it with all other samples matching this generalized behavior. Currently analysts are heavily dependent upon exact signature matches that fail to detect even slight, automated changes to malware and do not allow for strong correlation across related samples.

This gap is rooted in the fact that systems are sufficiently expressive and flexible that there are many ways to conduct attacks and thus evade any specific detection approaches. There is also a semantic gap in that detectors are often seeking to detect malware using obfuscated bytes as opposed to the true malicious code. Malware is often permitted the freedom to decode itself after detection technologies have run, which allows each piece of malware to be unique and thus bypass signatures.

Research Direction

In the short term we need to improve de-obfuscation techniques as it provides a trivial way for malware to bypass detection and is required for analyzing the true payload of malware. A recent approach in this area provides a dynamic analysis engine that forces a binary to execute by flipping predicates at conditional statements and monitoring for crashes [16]. It is neither sound nor complete but may be one option for handling a large subset of binaries.

In the medium term automated ways to identify malicious behaviors are needed, something that is primarily a manual process today. One promising approach automatically extracts malicious behavior from benign and malicious samples using graph mining and concept analysis techniques [27]. Long term efforts should examine approaches that are less reliant upon specifics of attacks, such as data flow analysis and policy violation detection.

Gap: Inability to predict attacks

Likelihood: M Impact: M Cost: M

Description

There is currently minimal capability for predicting the targets and types of targets. The ability to predict attacks would enable more timely detection, better collection of information, and ultimately more thorough attack analyses. Researchers and developers could build more applicable detection tools and analysts could configure sensors to provide more focused logging. Unfortunately any existing prediction approaches are limited to known attacks, similar networks, or require extensive human specification of possible attacker actions [21] [69]. The root of these challenges arises because of the large number of ways to attack systems and the variations in systems and networks. Additionally, the enormous volume of baseline network traffic limits or eliminates more computationally intensive analyses.

Research Direction

Useful predictions may take significant research effort but the most immediate results may be realized by focusing on predicting identical systems or systems running identical software before moving onto predicting attacks on more dissimilar systems and networks. One promising approach is to detect websites that might soon be compromised based upon past compromises at similar sites [79].

In the mid to long term research should be undertaken that improves the ability to predict types of attacks rather than simply the targets of attacks.

Long term there is a more fundamental game-theoretic problem that if the attacker expects defenders to predict certain actions the attacker will take a different action. Approaches must therefore also be developed to conceal predictions and to confuse the adversary.

Gap: Lack well-accepted attack taxonomies

Likelihood: L Impact: L Cost: L

Description

Improving the analysis of attacks requires well-developed taxonomies that capture the steps, requirements, and dependencies of attack components. The lack of such taxonomies results in analysts and researchers creating informal mental models of attacks, reducing quality and increasing overall defensive effort due to a lack of sharing. Decision makers also have limited ability to predict potential results of investments in a given area due to the lack of structured ways for understanding how that area impacts related components of attacks. Network designers and operators also require attack taxonomies to aid with determining sensor placement and to know what information about an attack could be captured at each vantage point. The expressiveness and flexibility of our systems has enabled attackers to accomplish goals in an extremely large number of ways and this has complicated the creation of, and agreement upon taxonomies.

Research Direction

Overly general taxonomies have proven challenging to create and have not seen widespread adoption. To achieve more useful results in the short term, research should be highly targeted to specific classes of attacks. Recent work described in [81] provides a strong foundation for establishing a taxonomy for memory corruption attacks by presenting a model of steps, requirements, and dependencies for these types of attacks. These highly targeted taxonomies can be broadened in the medium term. The long term objective is to incorporate the taxonomies from various attack areas into more general taxonomies while maintaining utility.

4.4 PRODUCTION AND ANALYSIS ENVIRONMENT ASYMMETRY

Gap: Lack of data sets to evaluate the ground truth

Likelihood: H Impact: H Cost: L

Description

Data sets are required for assessing the effectiveness of attack analysis technologies, comparing across technologies, and evaluating potential improvements. The lack of appropriate data sets remains a substantial problem in the security community in general, and is particularly pronounced in malware detection [73]. Notably, many technologies cannot be fully evaluated because we lack data sets with ground truth. Historical samples are traditionally used, which are often not representative of future attacks. Further, the ground truth in such data sets is typically labeled at a very high level, such as malware or not malware, prohibiting more granular evaluations. Organizations that have higher fidelity data sets are often discouraged from making them available due to concerns about revealing sensitive information to the public while retaining meaningful information for analysis [28]. This gap hinders researchers evaluating potential detection mechanisms, developers of detection models, and those computing risk.

The main cause for the lack of data sets is that we do not know the relevant data from our production environments that should be captured. Additionally, more useful data is often left underexplored as we lack means for properly anonymizing our data and thus are unable to share the best data sets.

Research Direction

Short term efforts are needed to build more representative, current, and transparent data sets. The work described in [73] can serve as a starting point for better understanding current deficiencies and provides some specific suggestions for improving malware experiments. The data sets should be more realistic and more dynamic, potentially utilizing a sliding window that incorporates more recent data and phases out irrelevant data [89].

In the medium term research is needed to improve privacy-preserving approaches for anonymizing data sets in order to encourage publication and sharing [28]. In addition to improving anonymity this research must simultaneously consider the extent to which data attributes necessary for evaluating detection technologies are preserved.

In the long term a stronger community should be created that focuses on data set creation. Attention must be given to making improved data sets widely available and providing guidance for the creation of new sets by the community.

Gap: Attack identification and analysis in isolated environments provides little fidelity

Likelihood: H Impact: M Cost: M

Description

Malware analysis often requires execution of a sample to observe its behavior (dynamic analysis) because of the limitations of static analysis. However, for safety and liability reasons the analysis environment has to be isolated from other networks (e.g. the Internet). This creates a strong tradeoff between the fidelity of the analysis and the isolation [32]. Many malware samples do not expose their malicious behavior unless they can connect to their C2 infrastructure. Moreover, by the time the analysis is conducted, it is possible that the C2 infrastructure does not exist or has been taken down. This is particularly a problem if the malware decrypts its payload only if it can receive a key from the C2 infrastructure. As a result, it is possible that dynamic analysis provides little fidelity for analyzing many samples because it is hard or impossible to provide the correct stimuli for the malware in an isolated environment. Note that even in the exact same environment many malware samples show different behaviors at different times [55]. This can be due to randomized behavior coded into their logic so that they can take different roles in the wild (e.g. a portion of them can become scanners for new victims, another portion can become C2 nodes, etc.).

The main root cause of this attack is the difference in the behavior of the analysis environment (an isolated system) and the production one (a system with connectivity to malware C2 infrastructure). Replicating the exact production environment for this purpose can be impossible to achieve since the C2 infrastructure may no longer be available. Another root cause is the semantic gap between that which is captured and understood by malware analysis platforms and the actual meaning of malware activity, such as higher-level communications with its C2 servers for unpacking and decrypting.

Research Direction

Short term research should focus on trying to emulate the expected environment of the malware using dummy protocol implementations or captured C2 traffic from other instances of the same malware [32]. Path exploration techniques may also be used to improve the fidelity, but they face the same challenges in isolation.

Medium term research should focus on automated sensors for capturing malware C2 communication for the purpose of repeatability of malware analysis and providing proper stimuli.

Long term research should focus on automated approaches for analyzing a large repository of captured C2 communication against a given malware sample in order to invoke and analyze the behavior of interest.

Gap: Malware analysis often lacks correctness, transparency, and realism

Likelihood: M Impact: M Cost: M

Description

Evaluating the effectiveness and accuracy of attack detection and analysis techniques often involves experimentation with live malware samples. However, many such studies in practice lack correctness, transparency, and realism [73]. Issues such as contamination with benign software, unbalanced distribution of malware samples over malware families, incorrect privilege levels for the analysis system (which allows evasion by the malware), ignoring malware artifacts, and unrealistic background activities often make such analysis incorrect. In addition, lack of information about malware families used, sample selection strategies, system configurations and connectivity, and the rationales for false positives/negatives and true positives hurt the transparency and repeatability of such analyses. Finally, problems with studying irrelevant malware families, difficulties with real-world (production environment) evaluation, generalization from a single configuration (e.g., generalizing Windows XP results to Windows 7), lack of proper stimuli for the malware, and lack of Internet connectivity makes such evaluation unrealistic.

The main root cause of this gap is that the environment that is used for malware analysis has a different behavior than the production environment in terms of its configurations, distributions of malware, and workloads, among other attributes. Other contributing factors are that our analysis is static (temporally and spatially) whereas malware is dynamic, and that there is usually a large volume of mostly benign activities that are missing in a malware analysis environment.

Research Direction

Short term research should focus on developing a corpus of malware samples that is 1) representative: it has samples from all current families, 2) dynamic: it frequently retires old samples, and 3) balanced: it has the correct distribution of malwares. The corpora should include proper tagging of samples based on their targeted environment and configurations.

Medium term research should focus on emulated approaches for providing the expected environment and stimuli for the malware under analysis.

Long term research should focus on generating realistic background activities or making the analysis environment look like the production one in order to achieve better realism and correctness.

Gap: Dynamic analysis techniques have limited coverage

Likelihood: M Impact: M Cost: M

Description

Dynamic analysis is a primary means for detecting malicious software as well as determining its behavior. In comparison with static analysis, it is a low-cost approach that requires less manual analysis and expertise. The major limitation is that it only provides coverage for a fraction of a program; specifically, it looks at the exact execution sequences encountered during a given run (or small set of runs) of a program. It is challenging to achieve high coverage even in benign applications, which often contain too many paths to feasibly evaluate. Dynamically analyzing malware is significantly more challenging because malware specifically employs tactics to prevent dynamic analyzers from reaching malicious paths by checking if it is executing in specific environments. Recent studies have claimed that as much as 81.4% of malware samples employ anti-virtualization protections [50]. Such protections are intentionally designed to prevent malware from executing malicious sequences while running inside virtual environments, which are commonly used for dynamic analysis. This gap in coverage prevents analysts from understanding malware without substantial investment in reverse engineering time and expertise, which is infeasible at scale given the sheer volume of malware. It also hinders researchers from better understanding and classifying malware.

The primary root cause is that it is hard to replicate the environment that is expected by the malware and thus trigger the dormant behavior. Thousands of environmental artifacts exist on any given system that malware can examine before triggering its payload, including software versions, CPU implementations [49], and timing measurements.

Research Direction

Short term efforts should develop automated ways of presenting environments to malware. For instance, ROZZLE [50] attempts to multi-execute JavaScript malware and follow all paths that are dependent upon environment-specific variables. Another promising approach attempts to reveal dormant behavior in browser extensions by running browser extensions on web pages specifically built to contain common elements of interest to malicious extensions [46].

Medium term efforts should design production environments to match analysis environments. While it is infeasible to replicate all possible environments, it is much more feasible to make a small set of known environments appear similar. With this approach the malware is forced to decide between the risk of exposing behavior to dynamic analyzers or missing an actual exploitation opportunity.

Longer term research needs to support creating sensors that build dynamic analysis into the hardware itself [26,49]. While hardware may indeed leave some artifacts for the malware to investigate, the surface is likely smaller than the current software-based sensors. Further, identical hardware could be deployed on both production and analysis systems, forcing the malware to decide between analysis evasion and infection rate.

Gap: Malware analysis often not repeatable

Likelihood: M   Impact: M   Cost: M

Description

Repeating previous executions of malicious software is important for analyzing the behavior of unknown malware and for performing malware experimentation. There is currently a gap in mimicking the initial execution of malware during any subsequent dynamic analyses. This prevents reverse engineers from fully understanding the impact of a malware infection and it hinders researchers in their efforts to evaluate malware analysis technologies, particularly those attempting to validate results. Repeatability of malware analysis is particularly hard because the infrastructure that controls the malware may be gone or taken down at the time of future analyses [32]. Repeating malware executions also introduces the challenge of containing the malware and preventing further damage.

The root of this gap stems from the fact that it is hard to replicate the initial infection environment. Infrastructure is volatile and the state of external infrastructure is often outside the control of the analysts, preventing exact repetition. This is exacerbated by the fact that malware is often designed to change its behavior with even slight changes to the environment [55]. Furthermore, protocols are commonly manipulated in non-standard ways [51] and custom protocols may even be used which require extensive manual reverse engineering effort in order to emulate.

Research Direction

Research should be directed towards emulating the infrastructure required by the malware such as network protocols, services, and hosts. In the short term focus on protocols that are custom-developed but widely used by malware. Improving the ability to fingerprint protocols is an important component for determining how widely used a given protocol is amongst malware. In the medium term it may be possible to automatically learn some protocols [32] in order to avoid the manual process of reverse engineering a protocol. Longer term, approaches are needed for emulating the victim hosts, which are often generic systems. In this way, the damage to victims can be limited if analysts must allow malware to contact malicious infrastructure for deeper analysis.

Gap: Machine learning effectiveness is fundamentally difficult to evaluate

Likelihood: L   Impact: L   Cost: H

Description

Machine learning techniques have been proposed in many cyber security contexts. However, the effectiveness of such approaches is very hard to evaluate since in many cases they learn on features that do not necessarily have strong ties to attacks. For example, approaches such as the one proposed for detecting malicious PDFs [77] learn on certain features such as file size, various header sizes, and offsets; however, these features do not necessitate maliciousness. The other side of this problem is that by changing these features, malware can evade being detected by ML-based approaches [89].

The main root cause of this gap is the difference between analysis environments and production ones. In production environments features used by the ML algorithms can take many different values while in the analysis environment they have limited values and they may be over-learned.

Another contributing root cause is the large volume of mostly benign activities that make ML difficult and hard to evaluate.

Research Direction

Medium term research should focus on ML approaches that operate under models that prevent simple evasion by adversaries [89]. This can be achieved by ensuring that the adversary cannot interfere with either the training phase or the classification phase of ML-based approaches.

Long term research can focus on extracting features that are necessitated by particular attack vectors, so that they cannot be perturbed by an attacker.

4.5 LEGACY SUPPORT REQUIREMENTS

Gap: Most UDP-based protocols are hard to analyze using stateless methods

Likelihood: M Impact: L Cost: M

Description

UDP-based protocols are prone to replay attacks because of IP spoofing. However, detecting such attacks can be very hard because it requires stateful analysis on the detection side while the protocol itself is stateless [72]. As a result, the analysis logic should be much more complex than the attack itself which creates unbalance in attacker's favor.

The main root cause of this gap is the requirement to support legacy protocols (e.g. UDP). These protocols where designed decades ago for a different environment, and the fact that modern systems still use and support them creates an uphill battle for detection of misuse attacks against them. Another contributing problem is the loose tying between identities and entities in the network that allows spoofing.

Research Direction

Short term research should focus on detecting and blocking spoof-enabled Autonomous Systems (ASes) across the network to prevent IP spoofing [53].

Medium term research should focus better tracking approaches that can avoid IP spoofing problems and analyzing trends and evolutions of different regions and subnets on the Internet.

Long term research should focus designing new network protocols that are more amenable to clustering and analysis of regions and prevent spoofing and anonymous redirection altogether.

4.6 SENSORS ARE STATIC SPATIALLY AND TEMPORALLY, WHILE ATTACKS ARE DYNAMIC

Gap: Attack identification/detection techniques often ignore important facts

Likelihood: H Impact: H Cost: M

Description

Attack identification/detection is hard in general. One way to tackle hard problems is to incorporate facts and ground truth in tackling the problem. Unfortunately, the vast majority of attack identification and detection capabilities ignore such facts [61]. Most existing identification and detection capabilities are built to work for generic attacks and vulnerabilities, while ignoring the skewed distributions and practicalities of such attacks and vulnerabilities. For example, most vulnerability analysis tools and techniques treat most vulnerabilities alike [66], or at best incorporate scores such as the CVSS scores. However, it has been shown that in reality, a small number of vulnerabilities account for the majority of attacks and that such scores are an inaccurate predictors of how widely the vulnerability will be used by attackers [61]. Skewed distributions in the wild relate to practicalities of exploiting certain vulnerabilities or mounting attacks, but existing detection and analysis capabilities often ignore them. Moreover, the notions of attack surface based on lines of code or input/output channels [56] are inaccurate in capturing actual vulnerability of a software application to attack because even exploited vulnerabilities have disproportionate impact on the attack surfaces in the wild [61]. These facts and ground truth information can also be used for prioritization of vulnerabilities for patching, targeted attack detection, resource allocation for analysis, etc.

The root cause of this problem is that sensors are static spatially and temporally. The facts that we rely on old snapshots of reality to make prioritization decisions (temporally static) and that we analyze attacks and vulnerabilities in test environment and generalize the results to every system (spatially static) contribute to this problem.

Research Direction

Short term research should focus on developing new metrics to prioritize attacks and vulnerabilities while incorporating their correct distributions and their actual impact on attack surfaces. A scoring system using the metrics developed in the related work [61] would be the first step to improve CVSS scores.

Medium term research should focus on developing targeted detection techniques that do not treat all attacks and vulnerabilities the same; rather, they incorporate ground truth data collected in the wild to detect different attacks with varying sensitivities and adjustable parameters. For example, if a vulnerability is being actively exploited, we can treat possible false positives with more strict actions. On the other hand, if a vulnerability is very hard to exploit, some false positives can be ignored.

Long term research should focus on automated scoring and prioritization systems based on observed traffics and detection events. Large-scale information sharing would be a necessary component of such an automated system.

Gap: Lack automated ways to build detection technologies from attacks

Likelihood: H Impact: M Cost: M

Description

Attacks are constantly changing, not only do new vulnerabilities appear every day but the same vulnerabilities can be exploited using many different payloads [62]. Current detection technologies have fixed update cycles that typically involve manual signature creation. One way to keep pace with constantly evolving attacks is to update our detection technologies by way of the attacks themselves. While in many cases this may require at least one system to be attacked, there is potential for the information learned from that attack to be useful for all other systems. Closing this gap could vastly limit the effectiveness of attacks to just a single system or a small group of system. The existence of this gap is a major challenge for those building detection technologies which spend tremendous amounts of manual effort to maintain them, it impacts network operators seeking to limit the damage to their network, as well as analysts.

The cause primarily lies in the fact that our detection technologies are largely static, updated at fixed patch points using specific signatures generated from manual analysis, as opposed to more continuous techniques that automatically incorporate the information learned from attacks. A secondary cause stems from the fact that we lack reversible or low impact defenses and thus are often hesitant to deploy automated strategies that might have significant operational ramifications. Detection technologies have an advantage in this regard in that one low impact course of action is to simply alert. This makes detection a strong candidate for automated approaches in comparison to other areas of defense.

Research Direction

In the short to mid term research is needed to identify potential candidate areas for detection technologies. The DARPA cyber grand challenge [24] is creating a completely automated capture-the-flag exercises where systems will automatically attack and defend one another. Expect research stemming from this program to provide insight into the feasibility and challenges of automated approaches.

Mid to long term research should seek to build automatic detection approaches in the identified candidate areas. Focus should initially be on using information about a small group of infected systems in order to generate detection information about similar systems that have yet to be infected. One promising approach [79] uses information from compromised web servers in order to detect similar sites that are also vulnerable and may soon be attacked. Other work has provided foundational ideas for automatically developing signatures and has attempted to automatically create signatures for worms that change their payload with each infection attempt in order to evade signature detection [62]. Another related research area that is in its early stages involves automatically generating patches for vulnerable C programs given a faulting program and a set of program test cases [90].

Gap: Analysis of IP address spaces to detect malicious subnets is static and incorrect

Likelihood: M Impact: L Cost: M

Description

Many network reputation analysis techniques rely on analysis of IP address regions to determine their maliciousness [52,87] in order to implement filters or blacklists. However, this type of analysis is often incorrect because it is static and has single granularity. For example, techniques that analyze IP regions at the given granularity (e.g. /8 addresses) assume that all machines in that region have the similar security postures. This may be true for a single organization (e.g. a university network), but it is not true for an ISP (e.g. different machines in AT&T /8 addresses can have very different postures). Moreover, since this type of analysis is often static, it assumes a fixed distribution over the entire region, whereas in reality attackers move and evolve in different regions which make the distribution time-dependent. As a result machine learning techniques based on such fixed distribution assumptions are not applicable in this context. Finally, the aggregation of smaller subnets into larger regions should be dynamic. A pre-determined aggregation strategy suffers from the above-mentioned problems [86].

The main root cause of this gap is that the analysis algorithms are static (e.g. predetermined aggregation, fixed granularity, etc.) while the maliciousness of an IP region is dynamic.

Research Direction

Short term research should focus on deploying already developed dynamic clustering techniques [86] to analyze IP regions in a more flexible way.

Medium term research should focus better tracking approaches that can avoid IP spoofing problems and analyzing trends and evolutions of different regions and subnets on the Internet.

Long term research should focus designing new network protocols that are more amenable to clustering and analysis of regions and prevent spoofing and anonymous redirection altogether.

4.7 LOOSE TYING BETWEEN IDENTITIES AND ENTRIES OVER THE NETWORK

Gap: Reputation metrics are insufficient to prevent botnets from operating

Likelihood: H   Impact: M   Cost: M

Description

Botnet operators have a large number of evasion and obfuscation techniques that prevent reputation metrics from becoming sufficient data sources for preventing botnets from operating at high efficiency [40]. Reputation metrics are still useful information sharing databases to incur maintenance costs upon botnet operators and to help network defenders cleanup their networks. However, tier 3 threat actors can easily invest in botnet evasion and obfuscation techniques to create new botnets after they have derived their value out of older and detected botnets.

Research Direction

Nazca [40] demonstrates a system that finds botnets via behavioral characteristics on the network. The key insight is that botnet operators wish to install their botnet on the largest population possible. Nazca works at the ISP level and focuses on detecting the malware installation stage via network calculated aggregates. The malware installation is focused on determining whether or not executable/binary download request over HTTP share traits common with malware downloads as opposed to benign software downloads. For example one trait is whether or not a binary being downloaded is downloaded from a large number of unique URLs, which is a common detection evasion technique.

Future research in detecting botnets follows the research direction of detecting behaviors that are common to malware and not common to other applications. Long-term this will result in an enumeration of malware behavioral characteristics, which will lead to the discovery of a core set of behaviors that must be performed in order to be malware. Perfect detection will be limited by whether or not these behaviors are exclusively used by threat actors. Without exclusive behavior, botnet detection will always be a cat and mouse game where botnet operators learn the detection thresholds of the defensive actors and tweak their techniques to evade the detectors. Defensive actors will pull samples of new botnet behavior to find the evasion and tweak their thresholds and metrics to continue to track and block such botnets. However, once a cat and mouse steady state is achieved we can start to measure whether or not botnet creation and maintenance remains cost effectives for tier 3 threat actors.

Gap: IP addresses cannot be trusted as identity on the Internet

Likelihood: H   Impact: H   Cost: H

Description

The internet was designed to be self-healing and to reliably deliver packets from end points to end points; it was not designed to verify the authenticity of packets. As such there exist numerous methods of faking identities on the internet. The internet as a whole continues to allow 2692 autonomous regions to spoof IP addresses [53]. Furthermore, control of IP space changes rather quickly over time [86], which means that adversaries can simply shift to new IP space with little cost. As such IP address cannot be used as a strong indicator of identity in order to determine who is responsible for an attack and therefore tracking campaigns of attacks is difficult.

Research Direction

Most attribution based research methods use additional sources of information to attribute identity to entities on the internet. One technique that has been used to some success is to use layer 7 HTTP fields to identify actors [96]. The idea is to compute fingerprints that are derived from HTTP fields in order identify a network actor who may change their network layer identity but not change their applications. This technique uses the user-agent string combined with IP addresses to identify common actors. For research directions this work highlights the feasibility of using layer 7 information to fingerprint the application and thus the actor that generated the traffic. Each type of common internet protocol (e.g., HTTP, DNS, NTP) could be investigated for statistically significant fields that can be used to derive the fingerprints of each adversary's specific applications and techniques. If such fields can be located, adversaries will find it more difficult to fake their identity.

Gap: DNS is not secure; DNS can be used to bypass most protocols

Likelihood: H Impact: M Cost: H

Description

The domain name service (DNS) protocol is critical to the internet. As such, it cannot be turn off even though it was never designed to be secure, and it was designed to allow itself to be used in new and unique ways. There are numerous findings that the DNS protocol is abused to conduct malicious behavior. For example, DNS is often used to create covert and difficult to block command and control channels [29] as well as the source of many unexplainable DNS transactions. DNS has also been used to establish covert network tunnels because most access points rely upon DNS to implement security and payment redirection controls [4]. As a result, DNS remains a viable method conduct covert activities and bypass security mechanisms.

Research Direction

There are no short term solutions to this problem. Every internet protocol depends upon DNS to operate and because DNS servers are not centrally managed there is no mechanism to secure the behavior of DNS. More work in the spirit of the DNS analysis in [29] is required to determine what reforms to the DNS protocol are needed or discover method to separate malicious DNS activity from benign behavior.

Gap: Ability to evaluate security posture of Internet as a whole in order to predict attacks - defenders solely focused on individual networks

Likelihood: H Impact: M Cost: H

Description

The internet is a federation of independent actors. As such, each actor manages their own network, and they focus only on their own defense of their networks. However, most networks and service share common code bases and are vulnerable to the same types of attacks. Since each network is managed by a different actor, each network that shares a common vulnerability will be patched in the timeframe that each actor decided upon. Internet site that rely upon common code bases have no way to share their defensive postures with each other.

Research Direction

There are techniques that use data mining to determine site that have shared code base and use this information to predict which site on the internet will become compromised [14]. These techniques work because the data mining techniques uses feature such as the name of content management systems, which if vulnerable to attack become a good predictor of future compromise. Research in this direction should be focused on fingerprinting other sites as early warning indicators of possible comprise in the system that needs to be defended.

Gap: Different anti-virus/anti-malware software are inconsistent in malware labeling and family identification for the majority of samples (i.e. voting does not work)

Likelihood: L Impact: L Cost: M

Description

Malware corpora are organized by whoever constructs the corpus. Since there are a large number of malware variants, there is inconsistency in malware classification [40]. Recent studies show that the majority of antivirus and antimalware systems label the majority of malware samples inconsistently. As such, there is a need for a common method to classify malware.

Research Direction

Data mining techniques have been proposed to cluster malware according to anti-virus databases [67]. These techniques attempt to reconcile the different information presented by different anti-virus databases. However, an alternative trend in research is to derive behavioral specifications of malware [27]. This research has resulted in a tool called Holmes that records activities of malware to create a specification of that malware's behavior. Holmes has been evaluated for detecting new types of malware that is similar to previous samples of malware; however more research is required to determine if behavioral specifications can be used to identify new malware variants.

4.8 ENORMOUS VOLUME OF UNKNOWN ACTIVITIES (MOSTLY BENIGN)

Gap: Lack of standard analysis of memory corruption attacks

Likelihood: M Impact: H Cost: L

Description

Memory corruption bugs are one of the oldest problems in computer security. Applications written in low-level languages like C or C++ are prone to these kinds of bugs. The lack of memory safety (or type safety) in such languages enables attackers to exploit memory bugs by maliciously altering the program's behavior or even taking full control over the control-flow [81].

The common problem with memory corruption attacks is the enormous diversity in the effects of such attacks. As such, every memory corruption attack must be analyzed to determine its purpose and effect.

Research Direction

As long as security critical applications are written is memory unsafe languages, memory corruption attacks are an effective attack vector [81]. Furthermore, it is impossible to detect such attacks without secondary indicators such as system invariant violations [71]. Research should be focused on replacing applications that are written in memory unsafe languages with applications that are written in memory safe languages.

Gap: White list and black lists are not sufficient because they are mostly unmanageable

Likelihood: H Impact: M Cost: H

Description

White lists and black lists are common mechanisms used to detect when a protect system is communicating with an undesirable communicant. However, white lists become unmanageable when a networked system needs to be able to communicate with any non-hostile devices. Such situations are commonplace in the enterprise environment since most work computers need to access a variety of internet based services that change over time. It becomes difficult to manage an accurate black list for hostile actors on the Internet because malware authors may change their network identities faster than the black list can be updated [53].

Research Direction

Promising research directions to address the weaknesses of relying on white and black list involve using additional reputation source for different data sources to identify malware actors on the internet. One technique, CAMP [70], computes a reputation score for downloaded binaries to determine if the download server is malicious. This system is provided as a plug-in to Google Chrome and it uses Google's Safe Browsing API to detect known malicious binary by checking Google's list of known malicious binaries. For binaries that are not listed as malicious the plug-in rates extracts features from the download and the server behavior and sends this information to Google's service to provide additional behavioral checks for maliciousness.

Gap: Machine learning (ML) has an unacceptably high error rate for detecting attacks, and anomaly detection techniques cannot reliably detect attacks

Likelihood: L Impact: L Cost: M

Description

Machine learning techniques classify data according to precise questions and quality (validity) of the data set with which the algorithm is trained. In environment such as the internet, the variability of data sources and missions makes training an accurate classifier infeasible [80]. Furthermore, any positive match requires human investigation to determine whether or not the activity is an attack nor not; therefore, false positives, which are common in machine learning techniques, are costly in human asset time. Furthermore, research has shown that N-gram techniques are fundamentally incompatible with reliable attack detection [34]. As such, machine learning techniques cannot be relied upon to find undetected attacks.

Research Direction

Since automated machine learning techniques are ineffective at detecting unknown attacks, research should be directed to using automation to enable human analysts to quickly investigate network activity. Human intuition is most reliable method of detection malicious activity since automated methods are easily gamed and evaded [34, 80].

4.9 UNDERDEVELOPED REVERSIBLE AND LOW IMPACT DEFENSES

Gap: Zero-day exploit campaigns last on average almost a year; they are patched so slowly that they are weaponized by the attackers

Likelihood: H Impact: H Cost: L

Description

The average window of life-time for zero-day exploit once it has been revealed to the world is 312 days [11]. As such, any disclosed vulnerability becomes a real candidate for the short term exploitation of live vulnerable systems. Two current cases of such attempts for short term exploitation of systems have been recently observed with the HeartBleed [30], and ShellShock [85] vulnerabilities. Even when vulnerabilities are high profile, it is a sufficiently rewarding strategy for adversaries to attempt to exploit these vulnerabilities during the window of opportunity provided by the vulnerability.

Research Direction

There are no specific research programs that can fully alleviate the window of opportunity caused by zero-day disclosures. Current practice it to monitor protected system during the window of opportunity in order to detect if vulnerable resources have been attacked. All research that facilitates the monitoring of systems to detect and block remote exploitation (e.g., SNORT [22]) contribute useful tools for preventing adversaries from exploiting vulnerable systems and for determining the impact of successful attacks on vulnerable systems.

Gap: ML provides no method to make sense of detection of possible attacks

Likelihood: L Impact: L Cost: M

Description

Unless machine learning training sessions are conducted with rigor and preciseness, detectors that are generated by machine learning algorithm cannot explain why an attack is an attack [80]. As a result, humans must investigate an explain detector results at great cost.

Research Direction

Research in this area should be focused on creating detector that can be quickly verified by human operators. Any research that complicates the process of operators determining the cause of an attack detection should be eliminated.

4.10 SYSTEMS IMPLICITLY OR EXPLICITLY TRUST THE USERS

Gap: Almost all VM-based attack analysis techniques and virtual machine introspection (VMI) methods assume trust in the guest OS

Likelihood: L   Impact: L   Cost: L

Description

Virtual machine introspection (VMI) is a fundamental technique for safely executing and/or emulated untrusted code binaries for analysis. In order to execute this untrusted code, the appropriate guest operating system (OS) must be included in the VM machines execution image in order to correct emulate the execution of the untrusted binary. The current OSes have not been designed to be transparent to analysis and so all VMI techniques must either account for the behavior of the OS or risk allowing analysis upon untrusted binaries to become unsound [41].

This problem is commonly caused the semantic gap between the guest OS and the VMI technique. This gap stems from the fact that untrusted binaries can comprise and change the expected behavior of the operating system, and different versions of each OS will use different memory layout for key data structures, which complicate the construction of analysis tools. This semantic gap means that the analysis cannot trust their analysis of executed binary without analyzing the effect the binary has upon the guest OS, which may be an intractable endeavor.

Research Direction

There are no short term research directions to address this gap. Addressing this gap requires the redesign of modern operating systems such that each operating system has formal specification of their expected behavior. Work has been performed to create formally verified secure microkernels such as the seL4 project [63]; however, such kernel have not been specifically created to facilitate VMI.

Gap: Attackers can easily hide in ill-defined web standards; web-based attacks all arise from this problem (XSS, CSRF, etc.)

Likelihood: M   Impact: H   Cost: H

Description

Many web standard behaviors implicitly trust the referenced server. This enables many attacks against the users of web service such as cross site scripting attacks [44]. As such, web security depends heavily upon the web site to implement sanity checks and security mechanisms to protect users of websites.

Research Direction

There are proposed methods to refine site authentication methods using JavaScript that is served by the website to provide additional protection against unintended cross-site security references [44]. This method is a password based authentication method that is claimed to be secure by default. Investigation into whether or not critical website would benefit from the technique need to be performed.

This page intentionally left blank.

5. SUMMARY OF RESEARCH DIRECTIONS

In our gap analysis and evaluation of promising research directions, we have encountered seven high level areas of research applicable to multiple gaps that can improve the attack analysis capabilities significantly. In this section, we describe these high level directions. Note that the specific research directions have already been covered under each gap, and this section serves as a summary of the previous section.

POPULATION-BASED ATTACK ANALYSIS

Many attack analysis capabilities to date have focused on studying individual attacks; however, they suffer from the weaknesses and gaps covered in the previous section. Multiple promising research efforts that attempt to improve attack analysis capabilities shift the focus from individual attacks to studying the population of attacks form better fidelity and effectiveness. For example, in the malware analysis area, one such technique extracts the behavioral models of a population of malwares in order to better analyze a malware sample [40]. Another example of this general direction is analyzing communication patterns of a large population of nodes to detect botnets as opposed to detecting the exploitation attempt of the botnet on one node [33].

DYNAMIC SENSORS

Another general research direction to cover many of the gaps is using detection sensors that are more temporally and spatially dynamic. For example, numerous promising techniques have incorporated information from sensors placed on multiple parties involved in an attack (spatial dynamics) to detect attacks with better fidelity [43, 46]. In addition, other promising techniques have proposed temporal dynamics in the sensors. For example, one such technique changes the environment presented to malware to better expose its behavior [50] and another proposes a more temporally dynamic IP clustering and malicious subnet detection technique [86].

COLLABORATIVE ATTACK ANALYSIS

Many obfuscated attacks and sophisticated 0-day exploitation attempts may be impossible to detect completely for years to come. However, many promising research efforts suggest a more collaborative way to limit the damage of such attacks after the first few compromises. This would require information sharing among enterprises (e.g. USG or DoD systems). For example, if the malware's payload is obfuscated, as soon as it is de-obfuscated on a system and the malware exposes its behavior such information can be shared with other enterprises to provide them with better insights into that malware. Collaborative attack analysis can also significantly improve the accuracy of reputation-based metrics, provide better malware stimuli, and limit the damage of targeted attacks.

AUTOMATION

Another important research direction that can mitigate numerous gaps is incorporating more automation into currently manual processes. For example, automated signature-generation techniques, automated adjustment of sensor thresholds, and automated and revisable reaction capabilities can improve numerous gaps.

MULTI-RESOLUTION ANALYSIS

An important problem in many existing attack identification and detection capabilities is that they treat all vulnerabilities, attacks, and systems alike. A promising direction in various gaps is to make the analysis multi-resolution and dedicate more resources and in-depth analysis to more severe vulnerabilities, more common attack vectors, more sensitive parts of a code, and generally more sensitive parts of a system. Considering unbalanced distribution of these aspects in real world systems can help more effectively detect attacks with lower false positives.

REPRESENTATIVE CORPORA

Better taxonomies of attacks and more representative databases of attacks and malwares have been identified in many research efforts as a high priority short term direction. For example, representative malware databases must incorporate relevant malware, have balanced distribution of malware samples, have balanced distribution of malware families, and must incorporate proper mechanisms for retiring old and no longer relevant malware samples.

USING HARDWARE EXTENSIONS

Hardware support is missing in many areas of attack detection and prevention. However, many existing hardware extensions can be better utilized to improve the state of such capabilities in short term. For example, Intel Memory Protection Extension, Inter Software Guard Extension (SGX), Intel Trusted eXecution Technology (TXT), and in general hardware-assisted randomization and tagging can improve current attack detection capabilities significantly.

TABLE 3

Overall Gap Prioritization Table

Number	Gap	Likelihood	Impact	Cost
1	Static analysis techniques are hindered by packing and obfuscation techniques	H	H	L
2	Lack of data sets to evaluate the ground truth	H	H	L
3	Zero-day exploit campaigns last on average almost a year	H	H	L
4	Lack of basic understanding of information leakage attacks	M	H	L
5	Lack of standard analysis of memory corruption attacks	M	H	L
6	Lack of effective techniques for detecting code injection	H	H	M
7	Lack of effective techniques for detecting code reuse	H	H	M
8	Approaches based on syntactic information do not work	H	H	M
9	Attack identification/detection techniques often ignore important facts	H	H	M
10	Measures of attack surface and vulnerability scores are inaccurate and arbitrary	H	M	M
11	Attack identification and analysis in isolated environments provides little fidelity	H	M	M
12	Lack automated ways to build detection technologies from attacks	H	M	M
13	Reputation metrics are insufficient to prevent botnets from operating	H	M	M
14	Techniques to increase coverage suffer from path explosion	M	M	M
15	Inability to predict attacks	M	M	M
16	Malware analysis often lacks correctness, transparency, and realism	M	M	M
17	Dynamic analysis techniques have limited coverage	M	M	M
18	Malware analysis often not repeatable	M	M	M
19	Unable to generalize detection of malicious behavior	H	H	H
20	Lack well-accepted attack taxonomies	L	L	L

21	IP addresses cannot be trusted as identity on the Internet	H	H	H
22	Almost all virtual machine introspection (VMI) methods assume trust in the guest OS	L	L	L
23	Most UDP-based protocols are hard to analyze using stateless methods	M	L	M
24	Analysis of IP address spaces to detect malicious subnets is static and incorrect	M	L	M
25	Anti-virus systems provide an oracle to malware writers seeking to evade detection	H	M	H
26	Dearth of techniques focused on targeted attacks	M	H	H
27	DNS is not secure; DNS can be used to bypass most protocols	H	M	H
28	Difficulty to evaluate security posture of Internet as a whole	H	M	H
29	White list and black lists are not sufficient because they are mostly unmanageable	H	M	H
30	Attackers can easily hide in ill-defined web standards	M	H	H
31	Different anti-virus/anti-malware software are inconsistent	L	L	M
32	Machine learning (ML) has an unacceptably high error rate	L	L	M
33	ML provides no method to make sense of detection of possible attacks	L	L	M
34	Machine learning effectiveness is fundamentally difficult to evaluate	L	L	H

REFERENCES

[1] OWASP Top 10 List. https://www.owasp.org/index.php/Top_10_2013-Top_10, 2013.

[2] IBM Integrity Measurement Architecture, 2014.

[3] Sumayah A. Alrwais, Kan Yuan, Eihal Alowaisheq, Zhou Li, and XiaoFeng Wang. Understanding the Dark Side of Domain Parking. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 207–222, 2014.

[4] Analogbit. TCP-over-DNS tunnel software HOWTO. http://analogbit.com/tcp-over-dns_howto.

[5] Saswat Anand, Patrice Godefroid, and Nikolai Tillmann. Demand-Driven Compositional Symbolic Execution. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, pages 367–381, 2008.

[6] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. Innovative Technology for CPU Based Attestation and Sealing. Workshop on Hardware and Architectural Support for Security and Privacy, June 2013.

[7] John Aycock. A Brief History of Just-in-time. volume 35, pages 97–113, New York, NY, USA, June 2003. ACM.

[8] Elena Gabriela Barrantes, David H. Ackley, Trek S. Palmer, Darko Stefanovic, and Dino Dai Zovi. Randomized Instruction Set Emulation to Disrupt Binary Code Injection Attacks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS 2003, Washington, DC, USA, October 27-30, 2003*, pages 281–289, 2003.

[9] Adam Barth, Collin Jackson, and John C. Mitchell. Robust Defenses for Cross-Site Request Forgery. In *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*, pages 75–88, 2008.

[10] Leyla Bilge, Davide Balzarotti, William K. Robertson, Engin Kirda, and Christopher Kruegel. Disclosure: Detecting Botnet Command and Control Servers Through Large-Scale NetFlow Analysis. In *28th Annual Computer Security Applications Conference, ACSAC 2012, Orlando, FL, USA, 3-7 December 2012*, pages 129–138, 2012.

[11] Leyla Bilge and Tudor Dumitras. Before We Knew It: An Empirical Study of Zero-day Attacks in the Real World. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 833–844, New York, NY, USA, 2012. ACM.

[12] Stevens Le Blond, Adina Uritesc, Cédric Gilbert, Zheng Leong Chua, Prateek Saxena, and Engin Kirda. A Look at Targeted Attacks Through the Lense of an NGO. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 543–558, 2014.

[13] Nathaniel Boggs, Sharath Hiremagalore, Angelos Stavrou, and Salvatore J. Stolfo. Cross-Domain Collaborative Anomaly Detection: So Far Yet So Close. In *Recent Advances in Intrusion Detection - 14th International Symposium, RAID 2011, Menlo Park, CA, USA, September 20-21, 2011. Proceedings*, pages 142–160, 2011.

[14] Kevin Borgolte, Christopher Kruegel, and Giovanni Vigna. Delta: Automatic Identification of Unknown Web-based Infection Campaigns. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer &#38; Communications Security*, CCS '13, pages 109–120, New York, NY, USA, 2013. ACM.

[15] Erik Bosman, Asia Slowinska, and Herbert Bos. Minemu: The World's Fastest Taint Tracker. In *Recent Advances in Intrusion Detection - 14th International Symposium, RAID 2011, Menlo Park, CA, USA, September 20-21, 2011. Proceedings*, pages 1–20, 2011.

[16] Rodrigo Rubira Branco, Gabriel Negreira Barbosa, and Pedro Drimel Neto. Scientific but Not Academical Overview of Malware Anti-Debugging, Anti-Disassembly and Anti-VM Technologies. *Black Hat*, 2012.

[17] Tom Brosch and Maik Morgenstern. Runtime Packers: The Hidden Problem? *Black Hat*, 2006.

[18] Nicholas Carlini and David Wagner. ROP is Still Dangerous: Breaking Modern Defenses. In *Proceedings of the 23rd USENIX Conference on Security Symposium*, SEC'14, pages 385–399, Berkeley, CA, USA, 2014. USENIX Association.

[19] Stephen Checkoway, Lucas Davi, Alexandra Dmitrienko, Ahmad-Reza Sadeghi, Hovav Shacham, and Marcel Winandy. Return-oriented Programming Without Returns. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 559–572, New York, NY, USA, 2010. ACM.

[20] Xiaobo Chen, Mike Scott, and Dan Caselden. New Zero-Day Exploit Targeting Internet Explorer Versions 9 Through 11 Identified in Targeted Attacks, April 2014.

[21] Casey Cipriano, Ali Zand, Amir Houmansadr, Christopher Kruegel, and Giovanni Vigna. Nexat: A History-Based Approach to Predict Attacker Actions. In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, pages 383–392, New York, NY, USA, 2011. ACM.

[22] Cisco. https://www.snort.org/.

[23] Johannes Dahse and Thorsten Holz. Static Detection of Second-Order Vulnerabilities in Web Applications. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 989–1003, 2014.

[24] DARPA. Cyber Grand Challenge. http://www.darpa.mil/cybergrandchallenge/, 2014.

[25] Lucas Davi, Ahmad-Reza Sadeghi, and Marcel Winandy. Dynamic Integrity Measurement and Attestation: Towards Defense Against Return-oriented Programming Attacks. In *Proceedings*

of the 2009 ACM Workshop on Scalable Trusted Computing, STC '09, pages 49–54, New York, NY, USA, 2009. ACM.

[26] Brendan Dolan-Gavitt, Tim Leek, Josh Hodosh, and Wenke Lee. Tappan Zee (North) Bridge: Mining Memory Accesses for Introspection. In *Proceedings of the 2013 ACM SIGSAC conference on Computer &#38; communications security*, CCS '13, pages 839–850, New York, NY, USA, 2013. ACM.

[27] Matt Fredrikson, Somesh Jha, Mihai Christodorescu, Reiner Sailer, and Xifeng Yan. Synthesizing Near-Optimal Malware Specifications from Suspicious Behaviors. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 45–60, Washington, DC, USA, 2010. IEEE Computer Society.

[28] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving Data Publishing: A Survey of Recent Developments. *ACM Comput. Surv.*, 42(4):14:1–14:53, June 2010.

[29] Hongyu Gao, Vinod Yegneswaran, Yan Chen, Phillip Porras, Shalini Ghosh, Jian Jiang, and Haixin Duan. An Empirical Reexamination of Global DNS Behavior. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 267–278, New York, NY, USA, 2013. ACM.

[30] Christopher Glyer. Attackers Exploit the Heartbleed OpenSSL Vulnerability to Circumvent Multi-factor Authentication on VPNs. Attackers Exploit the Heartbleed OpenSSL Vulnerability to Circumvent Multi-factor Authentication on VPNs, April 2014.

[31] David Grawrock. *Dynamics of a Trusted Platform: A Building Block Approach*. Intel Press, 1st edition, 2009.

[32] Mariano Graziano, Corrado Leita, and Davide Balzarotti. Towards Network Containment in Malware Analysis Systems. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pages 339–348, New York, NY, USA, 2012. ACM.

[33] Guofei Gu, Junjie Zhang, and Wenke Lee. BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2008, San Diego, California, USA, 10th February - 13th February 2008*, 2008.

[34] Dina Hadziosmanovic, Lorenzo Simionato, Damiano Bolzoni, Emmanuele Zambon, and Sandro Etalle. N-Gram against the Machine: On the Feasibility of the N-Gram Network Analysis for Binary Protocols. In *Research in Attacks, Intrusions, and Defenses - 15th International Symposium, RAID 2012, Amsterdam, The Netherlands, September 12-14, 2012. Proceedings*, pages 354–373, 2012.

[35] Seth Hardy, Masashi Crete-Nishihata, Katharine Kleemola, Adam Senft, Byron Sonne, Greg Wiseman, Phillipa Gill, and Ronald J. Deibert. Targeted Threat Index: Characterizing and Quantifying Politically-Motivated Targeted Malware. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 527–541, 2014.

[36] Mario Heiderich, Marcus Niemietz, Felix Schuster, Thorsten Holz, and Jörg Schwenk. Scriptless Attacks: Stealing the Pie Without Touching the Sill. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 760–771, New York, NY, USA, 2012. ACM.

[37] Jason Hiser, Anh Nguyen-Tuong, Michele Co, Matthew Hall, and Jack W. Davidson. ILR: Where'd My Gadgets Go? In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, pages 571–585, Washington, DC, USA, 2012. IEEE Computer Society.

[38] IDAPro. IDA: About. : https://www.hex-rays.com/products/ida/index.shtml.

[39] R. Intel. Introduction to Intel Memory Protection Extensions, 1672013, 2014.

[40] Luca Invernizzi, Stanislav Miskovic, Ruben Torres, Christopher Kruegel, Sabyasachi Saha, Giovanni Vigna, Sung-Ju Lee, and Marco Mellia. Nazca: Detecting Malware Distribution in Large-Scale Networks. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*, 2014.

[41] Bhushan Jain, Mirza Basim Baig, Dongli Zhang, Donald E. Porter, and Radu Sion. SoK: Introspections on Trust and the Semantic Gap. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, SP '14, pages 605–620, Washington, DC, USA, 2014. IEEE Computer Society.

[42] Trevor Jim, J. Greg Morrisett, Dan Grossman, Michael W. Hicks, James Cheney, and Yanling Wang. Cyclone: A Safe Dialect of C. In *Proceedings of the General Track of the Annual Conference on USENIX Annual Technical Conference*, ATEC '02, pages 275–288, Berkeley, CA, USA, 2002. USENIX Association.

[43] Martin Johns, Björn Engelmann, and Joachim Posegga. XSSDS: Server-Side Detection of Cross-Site Scripting Attacks. In *Proceedings of the 2008 Annual Computer Security Applications Conference*, ACSAC '08, pages 335–344, Washington, DC, USA, 2008. IEEE Computer Society.

[44] Martin Johns, Sebastian Lekies, Bastian Braun, and Benjamin Flesch. BetterAuth: Web Authentication Revisited. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pages 169–178, New York, NY, USA, 2012. ACM.

[45] Min Gyung Kang, Pongsin Poosankam, and Heng Yin. Renovo: A Hidden Code Extractor for Packed Executables. In *Proceedings of the 2007 ACM Workshop on Recurring Malcode*, WORM '07, pages 46–53, New York, NY, USA, 2007. ACM.

[46] Alexandros Kapravelos, Chris Grier, Neha Chachra, Christopher Kruegel, Giovanni Vigna, and Vern Paxson. Hulk: Eliciting Malicious Behavior in Browser Extensions. In *Proceedings of the 23rd USENIX Conference on Security Symposium*, SEC'14, pages 641–654, Berkeley, CA, USA, 2014. USENIX Association.

[47] Gaurav S. Kc, Angelos D. Keromytis, and Vassilis Prevelakis. Countering Code-injection Attacks with Instruction-set Randomization. In *Proceedings of the 10th ACM Conference on*

Computer and Communications Security, CCS '03, pages 272–280, New York, NY, USA, 2003. ACM.

[48] James C. King. Symbolic Execution and Program Testing. *Commun. ACM*, 19(7):385–394, July 1976.

[49] Dhilung Kirat, Giovanni Vigna, and Christopher Kruegel. Barecloud: Bare-metal Analysis-based Evasive Malware Detection. In *Proceedings of the 23rd USENIX Conference on Security Symposium*, SEC'14, pages 287–301, Berkeley, CA, USA, 2014. USENIX Association.

[50] Clemens Kolbitsch, Benjamin Livshits, Benjamin Zorn, and Christian Seifert. Rozzle: De-cloaking Internet Malware. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, pages 443–457, Washington, DC, USA, 2012. IEEE Computer Society.

[51] Nupur Kothari, Ratul Mahajan, Todd D. Millstein, Ramesh Govindan, and Madanlal Musuvathi. Finding Protocol Manipulation Attacks. In *Proceedings of the ACM SIGCOMM 2011 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Toronto, ON, Canada, August 15-19, 2011*, pages 26–37, 2011.

[52] Balachander Krishnamurthy and Jia Wang. On Network-aware Clustering of Web Clients. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '00, pages 97–110, New York, NY, USA, 2000. ACM.

[53] Marc Kührer, Thomas Hupperich, Christian Rossow, and Thorsten Holz. Exit from Hell? Reducing the Impact of Amplification DDoS Attacks. In *Proceedings of the 23rd USENIX Conference on Security Symposium*, SEC'14, pages 111–125, Berkeley, CA, USA, 2014. USENIX Association.

[54] Volodymyr Kuznetsov, László Szekeres, Mathias Payer, George Candea, R. Sekar, and Dawn Song. Code-pointer Integrity. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, OSDI'14, pages 147–163, Berkeley, CA, USA, 2014. USENIX Association.

[55] Martina Lindorfer, Clemens Kolbitsch, and Paolo Milani Comparetti. Detecting Environment-sensitive Malware. In *Proceedings of the 14th International Conference on Recent Advances in Intrusion Detection*, RAID'11, pages 338–357, Berlin, Heidelberg, 2011. Springer-Verlag.

[56] Pratyusa K. Manadhata and Jeannette M. Wing. An Attack Surface Metric. *IEEE Transactions on Software Engineering*, 37(3):371–386, May 2011.

[57] Mirco Marchetti, Michele Messori, and Michele Colajanni. Peer-to-Peer Architecture for Collaborative Intrusion and Malware Detection on a Large Scale. In *Information Security, 12th International Conference, ISC 2009, Pisa, Italy, September 7-9, 2009. Proceedings*, pages 475–490, 2009.

[58] Ilias Marinos, Robert N. M. Watson, and Mark Handley. Network Stack Specialization for Performance. In *Twelfth ACM Workshop on Hot Topics in Networks, HotNets-XII, College Park, MD, USA, November 21-22, 2013*, pages 9:1–9:7, 2013.

[59] Ingo Molnar. Exec Shield, new Linux security feature. *News-Forge*, 2003.

[60] Greg Morrisett, David Walker, Karl Crary, and Neal Glew. From System F to Typed Assembly Language. *ACM Trans. Program. Lang. Syst.*, 21(3):527–568, May 1999.

[61] Kartik Nayak, Daniel Marino, Petros Efstathopoulos, and Tudor Dumitras. Some Vulnerabilities Are Different Than Others - Studying Vulnerabilities and Attack Surfaces in the Wild. In *Research in Attacks, Intrusions and Defenses - 17th International Symposium, RAID 2014, Gothenburg, Sweden, September 17-19, 2014. Proceedings*, pages 426–446, 2014.

[62] James Newsome and Dawn Xiaodong Song. Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2005, San Diego, California, USA*, 2005.

[63] NICTA. Seure Microkernel Project (seL4). http://ssrg.nicta.com/projects/seL4/ [Accessed 20 Oct 2014].

[64] Vasilis Pappas. kBouncer: Efficient and Transparent ROP Mitigation. *Columbia University*, 2012.

[65] Bryan Parno, Jonathan M. McCune, and Adrian Perrig. Bootstrapping Trust in Commodity Computers. In *31st IEEE Symposium on Security and Privacy, S&P 2010, 16-19 May 2010, Berleley/Oakland, California, USA*, pages 414–429, 2010.

[66] Shirley C. Payne. A Guide to Security Metrics. *SANS Institute Information Security Reading Room*, 2006.

[67] Roberto Perdisci and ManChon U. VAMO: Towards a Fully Automated Malware Clustering Validity Analysis. In *28th Annual Computer Security Applications Conference, ACSAC 2012, Orlando, FL, USA, 3-7 December 2012*, pages 329–338, 2012.

[68] Georgios Portokalidis and Angelos D. Keromytis. Fast and Practical Instruction-Set Randomization for Commodity Systems. In *Twenty-Sixth Annual Computer Security Applications Conference, ACSAC 2010, Austin, Texas, USA, 6-10 December 2010*, pages 41–48, 2010.

[69] Xinzhou Qin and Wenke Lee. Attack Plan Recognition and Prediction Using Causal Networks. In *Proceedings of the 20th Annual Computer Security Applications Conference*, ACSAC '04, pages 370–379, Washington, DC, USA, 2004. IEEE Computer Society.

[70] Moheeb Abu Rajab, Lucas Ballard, Noe Lutz, Panayiotis Mavrommatis, and Niels Provos. CAMP: Content-Agnostic Malware Protection. In *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*, 2013.

[71] Rice. Rice's theorem. http://en.wikipedia.org/wiki/Rice

[72] Christian Rossow. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*, 2014.

[73] Christian Rossow, Christian J. Dietrich, Chris Grier, Christian Kreibich, Vern Paxson, Norbert Pohlmann, Herbert Bos, and Maarten van Steen. Prudent Practices for Designing Malware Experiments: Status Quo and Outlook. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, pages 65–79, 2012.

[74] M. Schiffman, G. Eschelbeck, D. Ahmad, A Wright, and S. Romanosky. CVSS: A Common Vulnerability Scoring System. *NIAC*, 2004.

[75] Felix Schuster, Thomas Tendyck, Jannik Pewny, Andreas Maaß, Martin Steegmanns, Moritz Contag, and Thorsten Holz. Evaluating the Effectiveness of Current Anti-ROP Defenses. In *Research in Attacks, Intrusions and Defenses - 17th International Symposium, RAID 2014, Gothenburg, Sweden, September 17-19, 2014. Proceedings*, pages 88–108, 2014.

[76] Edward J. Schwartz, Thanassis Avgerinos, and David Brumley. All You Ever Wanted to Know about Dynamic Taint Analysis and Forward Symbolic Execution (but Might Have Been Afraid to Ask). In *31st IEEE Symposium on Security and Privacy, S&P 2010, 16-19 May 2010, Berleley/Oakland, California, USA*, pages 317–331, 2010.

[77] Charles Smutz and Angelos Stavrou. Malicious PDF Detection Using Metadata and Structural Features. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pages 239–248, New York, NY, USA, 2012. ACM.

[78] Robin Sommer and Anja Feldmann. NetFlow: Information Loss or Win? In *Proceedings of the 2nd ACM SIGCOMM Internet Measurement Workshop, IMW 2002, Marseille, France, November 6-8, 2002*, pages 173–174, 2002.

[79] Kyle Soska and Nicolas Christin. Automatically Detecting Vulnerable Websites Before They Turn Malicious. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 625–640, 2014.

[80] Robin Summer and Vern Paxson. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 305–316, Washington, DC, USA, 2010. IEEE Computer Society.

[81] Laszlo Szekeres, Mathias Payer, Tao Wei, and Dawn Song. SoK: Eternal War in Memory. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 48–62, 2013.

[82] P. Team. Pax address space layout Randomization (ASLR), 2003.

[83] Olivier Thonnard, Leyla Bilge, Gavin O'Gorman, Seán Kiernan, and Martin Lee. Industrial Espionage and Targeted Attacks: Understanding the Characteristics of an Escalating Threat. In *Proceedings of the 15th International Conference on Research in Attacks, Intrusions, and Defenses*, RAID'12, pages 64–85, Berlin, Heidelberg, 2012. Springer-Verlag.

[84] Thomas Toth and Christopher Kruegel. Accurate Buffer Overflow Detection via Abstract Payload Execution. In *Proceedings of the 5th International Conference on Recent Advances in Intrusion Detection*, RAID'02, pages 274–291, Berlin, Heidelberg, 2002. Springer-Verlag.

[85] J. Trost. Digging into ShellShock Exploitation attempts using ShockPot Data. https://www.threatstream.com/blog/shockpot-exploitation-analysis, September 2014.

[86] Shobha Venkataraman, David Brumley, Subhabrata Sen, and Oliver Spatscheck. Automatically Inferring the Evolution of Malicious Activity on the Internet. In *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*, 2013.

[87] Shobha Venkataraman, Subhabrata Sen, Oliver Spatscheck, Patrick Haffner, and Dawn Song. Exploiting Network Structure for Proactive Spam Mitigation. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, SS'07, pages 11:1–11:18, Berkeley, CA, USA, 2007. USENIX Association.

[88] Philipp Vogt, Florian Nentwich, Nenad Jovanovic, Christopher Kruegel, Engin Kirda, and Giovanni Vigna. Cross-Site Scripting Prevention with Dynamic Data Tainting and Static Analysis. In *Network and Distributed Systems Security Symposium (NDSS)*, 02 2007.

[89] Nedim Šrndic and Pavel Laskov. Practical Evasion of a Learning-Based Classifier: A Case Study. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, SP '14, pages 197–211, Washington, DC, USA, 2014. IEEE Computer Society.

[90] Westley Weimer, ThanhVu Nguyen, Claire Le Goues, and Stephanie Forrest. Automatically Finding Patches Using Genetic Programming. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 364–374, Washington, DC, USA, 2009. IEEE Computer Society.

[91] R. N. Wojtczuk. Advanced return-to-lib (c) exploits: PaX case study. *Phrack Magazine*, 2001.

[92] Jonathan Woodruff, Robert N. M. Watson, David Chisnall, Simon W. Moore, Jonathan Anderson, Brooks Davis, Ben Laurie, Peter G. Neumann, Robert Norton, and Michael Roe. The CHERI capability model: Revisiting RISC in an age of risk. In *ACM/IEEE 41st International Symposium on Computer Architecture, ISCA 2014, Minneapolis, MN, USA, June 14-18, 2014*, pages 457–468, 2014.

[93] Xiaobo. ASLR Bypass Apocalypse in Recent Zero-Day Exploits, October 2013.

[94] Fabian Yamaguchi, Nico Golde, Daniel Arp, and Konrad Rieck. Modeling and Discovering Vulnerabilities with Code Property Graphs. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, SP '14, pages 590–604, Washington, DC, USA, 2014. IEEE Computer Society.

[95] Jean Yang and Chris Hawblitzel. Safe to the Last Instruction: Automated Verification of a Type-Safe Operating System. In *Proceedings of the 2010 ACM SIGPLAN Conference on*

Programming Language Design and Implementation, PLDI 2010, Toronto, Ontario, Canada, June 5-10, 2010, pages 99–110, 2010.

[96] Ting-Fang Yen, Yinglian Xie, Fang Yu, Roger Peng Yu, and Martín Abadi. Host Fingerprinting and Tracking on the Web: Privacy and Security Implications. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*, 2012.

[97] Chenfeng Vincent Zhou, Christopher Leckie, and Shanika Karunasekera. A Survey of Coordinated Attacks and Collaborative Intrusion Detection. volume 29, pages 124–140, 2010.

This page intentionally left blank.